APPENDIX 1

(A Script)

---------- Generate new script ----------

0=generateStandardPreamble

1=generateText(Java EJB imports,Common.ini,random,vendor.csv)

2=generateText(com.testmybeans.vendor.Vendor imports,ctor.ini,random,vendor.csv)

3=generateText(TestThread header,Common.ini,random,vendor.csv)

4=generateText(TestThread.run implementation,Common.ini,random,vendor.csv)

5=generateText(com.testmybeans.vendor.Vendor
create,com.testmybeans.vendor.Vendor.ctor.ini,random,vendor.csv)

6=generateMethod(timingGetSet)

7=generateText(Bean end create,Common.ini,random,vendor.csv)

8=generateText(WebLogic getInitialContext,AppServer.ini,random,vendor.csv)

9=generateText(Close log,Common.ini,random,vendor.csv)

10=generateText(Log to disk,Common.ini,random,vendor.csv)

11=generateStandardTrailer

APPENDIX 2

(A Template)

```
[Java EJB imports]
//[Java EJB imports]
package com.testmybeans.client;
import javax.naming.InitialContext;
import javax.naming.Context;
import javax.naming.NamingException;
import java.rmi.RemoteException;
import java.util.*;
import java.io.*;

[Hello imports]
//[Hello imports]
import com.softbridge.hello.HelloHome;
import com.softbridge.hello.Hello;
import com.softbridge.hello.HelloPK;

[TestThread header]
//[TestThread header]
public class $system.programName$ extends Thread
{
int tempInt=0;
Double tempDouble=new Double(0.0);
Float tempFloat=new Float(0.0);
short tempShort=0;
long tempLong=0;
String tempString="";
long sTime=0;
long sStartTime=0;
int m_instanceNumber = 0;
static String m_urlName = null;
String threadNumber="";
public $system.programName$(int inst, String url)
{
m_instanceNumber = inst;
```

```
m_urlName = url;
threadNumber=Integer.toString(m_instanceNumber);
//System.out.println("Starting Instance: " + m_instanceNumber);
}

[TestThread.run implementation]
//[TestThread.run implementation]
public void run()
{
sTime = new Date().getTime();
sStartTime=sTime;
logError("elapsedStart", Long.toString(new Date().getTime()), "", "");

[Bean end create]
//[Bean end create]
}
catch (Exception e)
{
logError("except", "beanMethods", e.toString(), "");
}
finally
{
try
{
closeLog();
h.remove ();
h = null;
}
catch (Exception e)
{
logError("except", "closeHome", e.toString(), "");
}
}
}

[Bean end findByPrimaryKey]
```

```
//[Bean end findByPrimaryKey]
}
catch (Exception e)
{
logError("except", "beanMethods", e.toString(), "");
}
finally
{
try
{
closeLog();
h = null;
}
catch (Exception e)
{
logError("except", "closeHome", e.toString(), "");
}
}
}

[Log to disk]
//[Log to disk]
public void logError(String key, String functionName, String expected, String actual)
{
try
{
String elapsedTime = Long.toString(new Date().getTime() - sTime);
com.testmybeans.client.CThreadWriter.write(key + "~" + functionName + "~" +
expected + "~" + actual + "~" + elapsedTime + "~" + threadNumber);
}
catch (Exception e)
{
System.out.println("IOException: " + e.getMessage());
System.exit(1);
}
}
```

```
}

[Close log]
//[Close log]
public void closeLog()
{
sTime=sStartTime;
logError("elapsedEnd", Long.toString(new Date().getTime()), "", "");
}
```

APPENDIX 3
(Test Code)

```
#
//Bean under test: com.testmybeans.vendor.Vendor
//Author: TestMyBeans Code Generator v0.8
//Creation Date: Fri Dec 10 16:53:15 EST 1999
//Copyright (c) 1999 TestMyBeans, Inc. All rights reserved.
#
//[Java EJB imports]
package com.testmybeans.client;
import javax.naming.InitialContext;
import javax.naming.Context;
import javax.naming.NamingException;
import java.rmi.RemoteException;
import java.util.*;
import java.io.*;


//[TestThread header]
public class invoice extends Thread
{
  int tempInt=0;
  Double tempDouble=new Double(0.0);
  Float tempFloat=new Float(0.0);
  short tempShort=0;
  long tempLong=0;
  String tempString="";
  long sTime=0;
  long sStartTime=0;
  int m_instanceNumber = 0;
  static String m_urlName = null;
  String threadNumber="";
  public invoice(int inst, String url)
  {
    m_instanceNumber = inst;
    m_urlName = url;
```

```
    threadNumber=Integer.toString(m_instanceNumber);
    //System.out.println("Starting Instance: " + m_instanceNumber);
  }

  //[TestThread.run implementation]
  public void run()
  {
    sTime = new Date().getTime();
    sStartTime=sTime;
    logError("elapsedStart", Long.toString(new Date().getTime()), "", "");

    //[com.testmybeans.vendor.Vendor create]
    int msValue = (-100000) * ((int) new Date ().getTime ()) %
(Integer.MAX_VALUE/100000);
    int arg0 = m_instanceNumber + msValue;
    com.testmybeans.vendor.Vendor h = null;
    try
    {
      Context jndi = getInitialContext();
      com.testmybeans.vendor.VendorHome home =
(com.testmybeans.vendor.VendorHome) jndi.lookup("OEVendor");
      h = home.create(arg0);
    }
    catch (Exception e)
    {
      System.out.println ("com.testmybeans.vendor.VendorHome" + e.toString());
    }
    try
    {
      //--------com.testmybeans.vendor.Vendor(setCost)----------
      try
      {
        sTime = new Date().getTime();
        h.setCost(568);
        logError("SetTime", "setCost", "568", "");
      }
```

```
        catch (Exception e)
        {
        logError("except", "setCost", e.toString(), "");
        }
    //---------com.testmybeans.vendor.Vendor(setName)----------
    try
        {
        sTime = new Date().getTime();
        h.setName("qkgthfpw");
        logError("setTime", "setName", "qkgthfpw", "");
        }
    catch (Exception e)
        {
        logError("except", "setName", e.toString(), "");
        }
    //---------com.testmybeans.vendor.Vendor(setProduct)----------
    try
        {
        sTime = new Date().getTime();
        h.setProduct("dpnoolww");
        logError("setTime", "setProduct", "dpnoolww", "");
        }
    catch (Exception e)
        {
        logError("except", "setProduct", e.toString(), "");
        }
    //---------com.testmybeans.vendor.Vendor(getCost)----------
    try
        {
        sTime = new Date().getTime();
        if((tempInt = h.getCost()) != 555)
        logError("getFailed", "getCost", "555", Integer.toString(tempInt));
        else
        logError("getPassed", "getCost", "555", Integer.toString(tempInt));
        }
    catch (Exception e)
```

```
            {
                logError("except", "getCost", e.toString(), "");
            }
        //          com.testmybeans.vendor.Vendor(getName)
        try
            {
                sTime = new Date().getTime();
                if(!(tempString = h.getName()).equals("icgnhoje"))
                logError("getFailed", "getName", "icgnhoje", tempString);
                else
                logError("getPassed", "getName", "icgnhoje", tempString);
            }
        catch (Exception e)
            {
                logError("except", "getName", e.toString(), "");
            }
        //          com.testmybeans.vendor.Vendor(getProduct)
        try
            {
                sTime = new Date().getTime();
                if(!(tempString = h.getProduct()).equals("zwlhjoxk"))
                logError("getFailed", "getProduct", "zwlhjoxk", tempString);
                else
                logError("getPassed", "getProduct", "zwlhjoxk", tempString);
            }
        catch (Exception e)
            {
                logError("except", "getProduct", e.toString(), "");
            }
        //[Bean end create]
        }
    catch (Exception e)
        {
        logError("except", "beanMethods", e.toString(), "");
        }
    finally
```

```
        {
            try
            {
                closeLog();
                h.remove ();
                h = null;
            }
            catch (Exception e)
            {
                logError("except", "closeHome", e.toString(), "");
            }
        }
    }

    //[WebLogic getInitialContext]
    public static Context getInitialContext() throws javax.naming.NamingException
    {
        try
        {
            Properties p = new Properties();
            p.put(Context.INITIAL_CONTEXT_FACTORY,
"weblogic.jndi.T3InitialContextFactory");
            p.put(Context.PROVIDER_URL, m_urlName);
            return new javax.naming.InitialContext(p);
        }
        catch (Exception e)
        {
            System.out.println("getInitialContext Exception: "+e);
            System.exit(1);
        }
        return null;
    }
    //[Close log]
    public void closeLog()
    {
        sTime=sStartTime;
```

```
    logError("elapsedEnd", Long.toString(new Date().getTime()), "", "");
  }

  //[Log to disk]
  public void logError(String key, String functionName, String expected, String actual)
  {
    try
    {
      String elapsedTime = Long.toString(new Date().getTime() - sTime);
      com.testmybeans.client.CThreadWriter.write(key + "~" + functionName + "~" +
expected + "~" + actual + "~" + elapsedTime + "~" + threadNumber);
    }
    catch (Exception e)
    {
      System.out.println("IOException: " + e.getMessage());
      System.exit(1);
    }
  }
}
#
//Bean under test: com.testmybeans.vendor.Vendor
//Author: TestMyBeans Code Generator v0.8
//Creation Date: Fri Dec 10 16:53:15 EST 1999
//Copyright (c) 1999 TestMyBeans, Inc. All rights reserved.
#
//[Java EJB imports]
package com.testmybeans.client;
import javax.naming.InitialContext;
import javax.naming.Context;
import javax.naming.NamingException;
import java.rmi.RemoteException;
import java.util.*;
import java.io.*;


//[TestThread header]
```

```
public class invoice extends Thread
{
  int tempInt=0;
  Double tempDouble=new Double(0.0);
  Float tempFloat=new Float(0.0);
  short tempShort=0;
  long tempLong=0;
  String tempString="";
  long sTime=0;
  long sStartTime=0;
  int m_instanceNumber = 0;
  static String m_urlName = null;
  String threadNumber="";
  public invoice(int inst, String url)
  {
    m_instanceNumber = inst;
    m_urlName = url;
    threadNumber=Integer.toString(m_instanceNumber);
    //System.out.println("Starting Instance: " + m_instanceNumber);
  }

  //[TestThread.run implementation]
  public void run()
  {
    sTime = new Date().getTime();
    sStartTime=sTime;
    logError("elapsedStart", Long.toString(new Date().getTime()), "", "");

    //[com.testmybeans.vendor.Vendor create]
    int msValue = (-100000) * ((int) new Date ().getTime ()) %
(Integer.MAX_VALUE/100000);
    int arg0 =  m_instanceNumber  + msValue;
    com.testmybeans.vendor.Vendor h = null;
    try
    {
      Context jndi = getInitialContext();
```

```
        com.testmybeans.vendor.VendorHome home =
(com.testmybeans.vendor.VendorHome) jndi.lookup("OEVendor");
        h = home.create(arg0);
      }
    catch (Exception e)
      {
        System.out.println ("com.testmybeans.vendor.VendorHome" + e.toString());
      }
    try
      {
      //        com.testmybeans.vendor.Vendor(setCost) ----------
        try
          {
            sTime = new Date().getTime();
            h.setCost(568);
            logError("SetTime", "setCost", "568", "");
          }
        catch (Exception e)
          {
            logError("except", "setCost", e.toString(), "");
          }
      //        com.testmybeans.vendor.Vendor(setName) ----------
        try
          {
            sTime = new Date().getTime();
            h.setName("qkgthfpw");
            logError("setTime", "setName", "qkgthfpw", "");
          }
        catch (Exception e)
          {
            logError("except", "setName", e.toString(), "");
          }
      //        com.testmybeans.vendor.Vendor(setProduct) ----------
        try
          {
            sTime = new Date().getTime();
```

```
            h.setProduct("dpnoolww");
            logError("setTime", "setProduct", "dpnoolww", "");
        }
        catch (Exception e)
        {
            logError("except", "setProduct", e.toString(), "");
        }
        //---------com.testmybeans.vendor.Vendor(getCost)----------
        try
        {
            sTime = new Date().getTime();
            if((tempInt = h.getCost()) != 555)
            logError("getFailed", "getCost", "555", Integer.toString(tempInt));
            else
            logError("getPassed", "getCost", "555", Integer.toString(tempInt));
        }
        catch (Exception e)
        {
            logError("except", "getCost", e.toString(), "");
        }
        //---------com.testmybeans.vendor.Vendor(getName)----------
        try
        {
            sTime = new Date().getTime();
            if(!(tempString = h.getName()).equals("icgnhoje"))
            logError("getFailed", "getName", "icgnhoje", tempString);
            else
            logError("getPassed", "getName", "icgnhoje", tempString);
        }
        catch (Exception e)
        {
            logError("except", "getName", e.toString(), "");
        }
        //---------com.testmybeans.vendor.Vendor(getProduct)----------
        try
        {
```

```
            sTime = new Date().getTime();
            if(!(tempString = h.getProduct()).equals("zwlhjoxk"))
            logError("getFailed", "getProduct", "zwlhjoxk", tempString);
            else
            logError("getPassed", "getProduct", "zwlhjoxk", tempString);
            }
        catch (Exception e)
            {
            logError("except", "getProduct", e.toString(), "");
            }
        //[Bean end create]
        }
    catch (Exception e)
        {
        logError("except", "beanMethods", e.toString(), "");
        }
    finally
        {
        try
            {
            closeLog();
            h.remove ();
            h = null;
            }
        catch (Exception e)
            {
            logError("except", "closeHome", e.toString(), "");
            }
        }
    }

  //[WebLogic getInitialContext]
  public static Context getInitialContext() throws javax.naming.NamingException
    {
    try
        {
```

```
        Properties p = new Properties();
        p.put(Context.INITIAL_CONTEXT_FACTORY,
"weblogic.jndi.T3InitialContextFactory");
        p.put(Context.PROVIDER_URL, m_urlName);
        return new javax.naming.InitialContext(p);
      }
    catch (Exception e)
      {
        System.out.println("getInitialContext Exception: "+e);
        System.exit(1);
      }
    return null;
  }
  //[Close log]
  public void closeLog()
  {
    sTime=sStartTime;
    logError("elapsedEnd", Long.toString(new Date().getTime()), "", "");
  }

  //[Log to disk]
  public void logError(String key, String functionName, String expected, String actual)
  {
    try
      {
        String elapsedTime = Long.toString(new Date().getTime() - sTime);
        com.testmybeans.client.CThreadWriter.write(key + "~" + functionName + "~" +
expected + "~" + actual + "~" + elapsedTime + "~" + threadNumber);
      }
    catch (Exception e)
      {
        System.out.println("IOException: " + e.getMessage());
        System.exit(1);
      }
  }
}
```

Claims with changes shown:

1. (Amended) A system for determining performance of ~~an~~ technology based software component of an application under test in response to load, the system comprising:

a) coordination software;

b) at least one code generator, receiving as an input commands from the coordination software and having as an output client test code;

c) at least one test engine, receiving as an input commands from the coordination software, the test engine comprising a computer server having at least one ~~JVM~~ software implementation of a processor executing at least one instance of the client test code;

d) at lease one data log having computerized memory, the memory holding timing data created by the instances of the client test code ; and

e) at least one data analyzer software, operatively connected to the data log, having an output that represents performance of the software component of the application under test in response to load.

2. (Amended) The system of claim 1 wherein said at least one ~~JVM~~ software implementation of a processor executes multiple threads, each thread comprising an instance of the client test code.

6. (Amended) The system of claim 2 wherein said at least one ~~JVM~~ software implementation of a processor is synchronized to start execution of an instance of the client test code with another of said at least one ~~JVM~~ software implementation of a processor about to start execution an instance of the client test code.

7. (Amended) The system of claim 3 wherein the synchronization of at least one ~~JVM~~ software implementation of a processor to another of said at least one ~~JVM~~ software implementation of a processor is performed independently of the time set on each system.

8. (Amended) The system of claim 3 wherein said at least one ~~JVM~~ software implementation of a processor is set to start execution of the client test code a predetermined time after another of said at least one ~~JVM~~ software implementation of a processor is set to start execution of the test client code.

6. (Amended) The system of claim 2 wherein said at least one ~~JVM~~ software implementation of a processor is set to start execution of the client test code independent of another of said at least one ~~JVM~~ software implementation of a processor set to start execution of the client test code.

7. (Amended) A computer program product for determining performance of a technology based software component of an application under test in response to load, the computer program product comprising a computer usable medium having computer readable code thereon, including program code comprising:

a) instructions for coordination software;

b) instructions for at least one code generator, receiving as an input commands from the coordination software and having as an output client test code;

c) instructions for at least one test engine, receiving as an input commands from the coordination software, the test engine comprising a computer server having at least one ~~JVM~~ software implementation of a processor executing at least one instance of the client test code;

d) instructions for providing at lease one data log having computerized memory, the memory holding timing data created by the instances of the client test code; and

e) instructions for providing at least one data analyzer, operatively connected to the data log, having an output that represents performance of ~~the~~ the software component of the application under test in response to load.

8. (Amended) The computer program product of claim 7 further comprising instructions for causing said at least one ~~JVM~~ software implementation of a processor to

execute multiple threads, each thread comprising an instance of the client test code.

12. (Amended) The computer program product of claim 7 further comprising instructions for causing said at least one ~~JVM~~ software implementation of a processor to be synchronized to start execution of an instance of the client test code with another of said at least one ~~JVM~~ software implementation of a processor about to start execution an instance of the client test code.

13. (Amended) The computer program product of claim 9 further comprising instructions wherein the synchronization of at least one ~~JVM~~ software implementation of a processor to another of said at least one ~~JVM~~ software implementation of a processor is performed independently of the time set on each system.

14. (Amended) The computer program product of claim 9 further comprising instructions wherein said at least one ~~JVM~~ software implementation of a processor is set to start execution of the client test code a predetermined time after another of said at least one ~~JVM~~ software implementation of a processor is set to start execution of the test client code.

12. (Amended) The computer program product of claim 8 further comprising instructions wherein said at least one ~~JVM~~ software implementation of a processor is set to start execution of the client test code independent of another of said at least one ~~JVM~~ software implementation of a processor set to start execution of the client test code.

13. (Amended) A method of testing a computerized application, the application under test having a plurality of technology based software components, at least one component having at least one component method therein, the method of testing comprising the steps of:

a) providing test code to exercise ~~a~~said software component;

b) creating a plurality of copies of the test code;

c) simultaneously executing the plurality of copies of the test code;

g) providing a folder for each component method of the software component being exercised;

h) recording times for each component method of the software component being exercised; and

i) analyzing the recorded times to present information on performance of each component method of the software component being exercised.

14. (Amended) The method of claim 13 wherein the software components are selected from Enterprise Java Beans technology based softweare components and Component Object Module technology based software componentss.

19. (Amended) The method of claim 13 wherein said step of recording times further comprises recording said times for each component method in a respective folder for said component method.

20. (Amended) The method of claim 15 wherein each folder is used to provide calculations for said component method from said times recorded in the folder.

21. The method of claim 16 wherein said calculations are selected from the group consisting of the average response time of the items within the folder, and the total response time of the items within the folder.

22. (Amended) A computer program product for testing a computerized application, the application under test having a plurality of technology based software components, at least one software component having at least one component method therein, the computer program product comprising a computer usable medium having computer readable code thereon, including program code comprising:

a) instructions for providing test code to exercise a software component;

h) instructions creating a plurality of copies of the test code;

i) instructions for simultaneously executing the plurality of copies of the test code;

j) instructions for providing a folder for each component method of the software component being exercised;

k) instructions for recording times for each component method of the software component being exercised; and

f) instructions for analyzing the recorded times to present information on performance of each component method of the software component being exercised.

19. (Amended) The computer program product of claim 18 wherein the software components are selected from Enterprise Java Bean technology based software componentss and Component Object Modules technology based software components.

22. (Amended) The computer program product of claim 18 further comprising instructions for recording said times for each component method in a respective folder for said component method.

23. (Amended) The computer program product of claim 20 further comprising instructions for providing calculations for said component method from said times recorded in the folder.

22. The computer program product of claim 21 for causing said calculations to be selected from the group consisting of the average response time of the items within the folder, and the total response time of the items within the folder.

23. (Amended) A system for determining performance of a technology based software component of an application under test in response to load, the system comprising:

a) coordination software;

b) at least one code generator, receiving as an input commands from the coordination software and having as an output client test code, said code generator providing a template for a datatable, said datatable used to provide information for exercising the software component of the application under test;

c) at least one test engine, receiving as an input commands from the coordination software, the test engine comprising a computer server having a plurality of threads thereon, each thread executing an instance of the client test code;

l) at leaset one data log having computerized memory, the memory holding timing data created by the instances of the client test code in the plurality of threads; and

m) at least one data analyzer software, operatively connected to the data log, having an output that represents performance of the software component of the application under test in response to load.

24. The system of claim 23 wherein said datatable includes a plurality of rows and a plurality of columns wherein said columns are used for parameters and said rows represent users.

25. The system of claim 23 wherein said datatable is in a .CSV format.

26. The system of claim 23 wherein when said datatable contains fewer rows than the number of virtual users provided by said test engine, then said test code will cycle through said data table and then start over beginning with the first row of said datatable.

27. (Amended) A computer program product for determining performance of a technology based software component of an application under test in response to load, the computer program product comprising a computer usable medium having computer readable code thereon, including program code comprising:

a) instructions for coordination software;

b) instructions for at least one code generator, receiving as an input commands from the coordination software and having as an output client test code, said code generator providing a template for a datatable, said datatable used to provide information for exercising the software component of the application under test;

c) instructions for at least one test engine, receiving as an input commands from the coordination software, the test engine comprising a computer server having a

plurality of threads thereon, each thread executing an instance of the client test code;

d) instructions for providing at lease one data log having computerized memory, the memory holding timing data created by the instances of the client test code in the plurality of threads; and

e) instructions for providing at least one data analyzer software, operatively connected to the data log, having an output that represents performance of the software component of the application under test in response to load.

28. The computer program product of claim 27 wherein said datatable includes a plurality of rows and a plurality of columns wherein said columns are used for parameters and said rows represent users.

29. The computer program product of claim 27 wherein said datatable is in a .CSV format.

30. The computer program product of claim 28 wherein when said datatable contains fewer rows than the number of virtual users provided by said test engine, then said test code will cycle through said data table and then start over beginning with the first row of said datatable.

31. (Amended) A method of testing a computerized application under test that allows simultaneous users over a computer network, the application under test having a plurality of technology based software components, the method of testing comprising the steps of:

a) providing test code to exercise the software a component, said software component including at least one component method;

b) providing a class file for each component method of said software component directly to each user;

c) creating a first plurality of copies of the test code;

d) simultaneously executing the first plurality of copies of test code while recording times between events in each of the first plurality of copies of test code;

e) creating a second plurality of copies of test code,

f) simultaneously executing the second plurality of copies of test code while recording times between events in each of the second plurality of copies of test code;

g) repeating a predetermined number of times the steps of creating plural copies of the test code and simultaneously executing the plural copies while recording event times; and

h) analyzing the recorded times to present information on the performance of the software component of the application under test as a function of load.

32. The method of claim 31 wherein said class file is provided as a compressed file.

33. (Amended) The method of claim 32 wherein said compressed file comprises a Java Archive technology based file.

34. (Amended) A computer program product for testing a technology based software component of a computerized application under test that allows simultaneous users over a computer network, the application under test having a plurality of software components, the computer program product comprising a computer usable medium having computer readable code thereon, including program code comprising:

instructions for providing test code to exercise a software component, said software component including at least one component method;

instructions for providing a class file for each component method of said software component directly to each user;

instructions for creating a first plurality of copies of the test code;

instructions for simultaneously executing the first plurality of copies of test code while recording times between events in each of the first plurality of copies of test code;

instructions for creating a second plurality of copies of test code;

instructions for simultaneously executing the second plurality of copies of test code while recording times between events in each of the second plurality of copies of test code;

instructions for repeating a predetermined number of times the steps of creating plural copies of the test code and simultaneously executing the plural copies while recording event times; and

instructions for analyzing the recorded times to present information on the performance of the software component of the application under test as a function of load.

35. The computer program product of claim 34 including instructions for causing said class file to be provided as a compressed file.

36. (Amended) The computer program product of claim 34 including instructions for causing said class file to be provided as a Java Archive technology based file.